RECEIVED
CENTRAL FAX CENTER

MAY 1 7 2006

Yee & Associates, P.C.

4100 Alpha Road
Suite 1100
Dallas, Texas 75244

Main No. (972) 385-8777
Facsimile (972) 385-7766

Facsimile Cover Sheet

To: Commissioner for Patents for **Examiner Satish Rampuria** **Group Art Unit 2191**	Facsimile No.: **571/273-8300**
From: Stephanie Fay, Legal Assistant to Betty Formby	No. of Pages Including Cover Sheet: **22**

Message:

Enclosed herewith:

- Transmittal of Appeal Brief; and
- Appeal Brief.

Re: Application No. **09/697,446**
Attorney Docket No: **AUS9-2000-0501-US1**

Date: Wednesday, May 17, 2006

Please contact us at (972) 385-8777 if you do not receive all pages indicated above or experience any difficulty in receiving this facsimile.

PLEASE CONFIRM RECEIPT OF THIS TRANSMISSION BY FAXING A CONFIRMATION TO 972-385-7766.

RECEIVED
CENTRAL FAX CENTER
MAY 17 2006

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Kumhyr et al.**	§	Group Art Unit: **2191**
	§	
Serial No.: **09/697,446**	§	Examiner: **Rampuria, Satish**
	§	
Filed: **October 26, 2000**	§	Attorney Docket No.: **AUS9-2000-0501-US1**
	§	
For: **Identifying Non-Externalized Text Strings That Are Not Hard-Coded**		

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

Certificate of Transmission Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being transmitted via facsimile to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, facsimile number (571) 273-8300 on May 17, 2006.
By: Stephanie Fay
Stephanie Fay

TRANSMITTAL OF APPEAL BRIEF

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:
ENCLOSED HEREWITH:

- Appeal Brief (37 C.F.R. 41.37)

A fee of $500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

Respectfully submitted,
Betty Formby
Betty Formby
Registration No. 36,536
AGENT FOR APPLICANTS

Duke W. Yee
Registration No. 34,285
ATTORNEY FOR APPLICANTS

YEE & ASSOCIATES, P.C.
P.O. Box 802333
Dallas, Texas 75380
(972) 385-8777

RECEIVED
CENTRAL FAX CENTER

MAY 1 7 2006

Docket No. AUS9-2000-0501-US1 ***PATENT***

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Kumhyr et al.**	§	
	§	Group Art Unit: **2191**
Serial No. **09/697,446**	§	
	§	Examiner: **Rampuria, Satish**
Filed: **October 26, 2000**	§	
	§	
For: **Identifying Non-Externalized Text Strings That Are Not Hard-Coded**	§ §	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

<u>Certificate of Transmission Under 37 C.F.R. § 1.8(a)</u>
I hereby certify this correspondence is being transmitted via facsimile to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, facsimile number (571) 273-8300 on May 17, 2006.

By: Stephanie Fay
Stephanie Fay

<u>APPEAL BRIEF (37 C.F.R. 41.37)</u>

This brief is in furtherance of the Notice of Appeal, filed in this case on April 10, 2006.

A fee of $500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

05/18/2006 TL0111 00000057 090447 09697446
01 FC:1402 500.00 DA

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-24

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: None
2. Claims withdrawn from consideration but not canceled: None
3. Claims pending: 1-24
4. Claims allowed: None
5. Claims rejected: 1-24
6. Claims objected to: None

C. CLAIMS ON APPEAL

The claims on appeal are: 1-24

Kurahyr et al. - 09/697,446

STATUS OF AMENDMENTS

No amendments have been submitted since the final office action was mailed.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

Claim 1 is directed to a method for identifying hard-coded strings for conversion (i.e., internationalization) (**Figure 2**, page 7, line 17 through page 8, line 20). The method contains the following steps:

- scanning programming code for a pair of delimiters used to delimit text strings (**210**, page 10, lines 3-15);
- determining whether the enclosed string is a path name to a resource file (**220**, **240**, page 10, line 17 through page 11, line 17); and
- if the enclosed string is not a path name to a resource file, the string is flagged as a possible hard-coded string (**270**, page 12, lines 13 through page 13, line 3).

B. CLAIM 2 - DEPENDENT

Claim 2 is directed to the alternative step that when the enclosed string is a path name to a resource file, the string is not flagged (**250**, page 11, line 22 through page 12, line 4).

C. CLAIM 5 - DEPENDENT

Claim 5 is directed to the path name being a resource bundle (page 9, lines 22-23).

D. CLAIM 6 - DEPENDENT

Claim 6 is directed to the string enclosed by the pair of delimiters being a path name to the resource file if the string is in a dot delimited notation (page 11, line 9).

E. CLAIM 9 - INDEPENDENT

Claim 9 is directed to a computer program product in a computer readable medium for identifying hard-coded strings that require conversion (not specifically shown, page 8, lines 21-23). This claim is a computer program analog to method claim 1.

F. CLAIM 17 - INDEPENDENT

Claim 17 is directed to a data processing system (**Figure 1**, reference **13**, page 7, line 17 through page 8, line 20) containing the following parts and means to perform the method of claim 1:

- a processor (**10**, page 7, lines 21-22);
- a memory (**14**, **16**, page 8, lines 3-6);
- an input mechanism (**24**, **26**, **28**, page 8, lines 15-16);
- an output mechanism (**38**, page 8, line 18);
- a bus system to couple the processor to the memory unit, input mechanism, and output mechanism (**12**, page 7, line 21 through page 8, line 18);
- means for scanning programming code for a pair of delimiters used to delimit text strings (**42**, page 7, line 24 through page 8, line 12);
- means for determining whether the enclosed string is a path name to a resource file (**42**, page 7, line 24 through page 8, line 12); and
- means for flagging the string as a possible hard-coded string if the enclosed string is not a path name to a resource file (**42**, page 7, line 24 through page 8, line 12).

G. CLAIM 18 - DEPENDENT

Claim 18 is directed to the data processing system of claim 17, additionally featuring means for not flagging said string as a possible hard-coded string if said string is a path name to said resource file (**42**, page 7, line 24 through page 8, line 12).

H. CLAIM 24 - DEPENDENT

Claim 24 is directed to the data processing system of claim 23, additionally featuring means for continuing to scan the programming code for additional pairs of delimiters if any programming code remains (**42**, page 7, line 24 through page 8, line 12).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. GROUND OF REJECTION 1 (Claims 1-24)

Claims 1-24 stand rejected under 35 U.S.C. § 102 as anticipated over **Bowen** et al., Keyword Searches of Structured Databases, U.S. Patent No. 6,094,649, July 25, 2000 (hereinafter "**Bowen**").

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1-24)

A.1. Claims 1-4, 7-12, 15-20, and 23-24

The rejection asserts that **Bowen** discloses the following from claim 1:

> - scanning programming code for a first pair of delimiters that are used to delimit text strings (col. 4, line 53-55 "a keyword search.., location identifier" and col. 7, lines 45-50 "...programming languages... Java... C++... and tools");
> - determining whether a string within said pair of string delimiters is a path name to a resource file (col. 4, line 56-62 "keyword with resource locators.., include. .URLs, hot links, file paths... among others"); and
> - if said string is not a path name to said resource file than flagging said string as a possible hard-coded string (col. 4 lines 22-26 "One method of the invention begins … selection ... one data, … in the structured database; each selected item ..., data and has a corresponding location identifier which identifies the item's location within the structured database").

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983).

Bowen does not anticipate the invention recited in claim 1 because **Bowen** does not identically show every element of the invention recited in this claim. More specifically, **Bowen** does not disclose "*scanning programming code for a first pair of string delimiters that are used to delimit text strings*"; nor does this reference disclose "*determining whether a string within said first pair of string delimiters is a path name to a resource file*". In reading the claim on **Bowen**, the rejection is interpreting claim 1 so broadly that the words lose their commonly understood meaning. Each of these steps will be discussed separately.

There are at least two ways in which **Bowen** does not meet the feature of "*scanning programming code for a first pair of string delimiters that are used to delimit text strings*": the reference is not scanning "*programming code*" and it is not scanning for "*delimiters*". In response to Appellants previous assertion that **Bowen** is not scanning code, the Examiner responded that "*Bowen discloses the keyword searching of highly structured data. Where data could be a document, flat-file etc. (col. 6, lines 55- 67). Bowen specifically explained that the invention is suited for programming languages such as Java, Pascal, C++, C CGI, Perl, SQL, APIs, SDKs, assembly, firmware, microcode, and/or other languages and tools. [quoted from Bowen, column 7, lines 48-50] These all are one skill in the art considered as programming languages*" (Final Office Action mailed January 10, 2006, page 3, paragraph 3). The cited excerpt reads as follows:

> As used here, a "structured database" is a collection of data items organized primarily by rules other than those governing natural languages such as English. The data items may contain natural language text such as addresses or part names in a relational database, but relations, tables, trees, or other structures are the primary means of organization. Structured database operations aid decision-making by allowing users to combine individual data items in various ways, as illustrated in the SQL query above.
>
> Relational databases are one example of structured databases; other examples include hierarchical, inverted-list, object-relational, object-oriented, and flat-file databases. Structured databases may be stored in a single location or distributed between several machines. Regardless of the approach taken to storage, many structured databases can be accessed through a network.

Bowen, column 6, line 55 through column 7, line 3

Additionally, the excerpt from **Bowen** that was quoted in the rejection, when read in context, states:

> Suitable software for implementing the invention is readily provided by those of skill in the art using the teachings presented here and programming languages and tools such as Java, Pascal, C++, C, CGI, Perl, SQL, APIs, SDKs, assembly, firmware, microcode, and/or other languages and tools.

Bowen, column 7, lines 45-50

These excerpts support the Examiner's assertion that **Bowen** is searching highly structured <u>data</u>, but not the assertion that **Bowen** is searching programming code. The programming code mentioned in this reference is the language in which the search engine of **Bowen** can be implemented; however, the programming code is not itself searched. Instead, the method of **Bowen** searches structured databases, exactly as its title suggests. **Bowen** does not show the searching of programming code.

The rejection further asserts that the string delimiters cited in claim 1 are shown in the following excerpt:

> Indexing may be accomplished by providing to a keyword search engine indexing agent both the textual representation of each selected item's data and the selected item's location identifier. The indexing agent produces an index that associates keywords with resource locators, and each resource locator includes a textual representation of a data item location identifier. Suitable indexing agents include web crawlers, indexing "bots", and other text indexing tools. Suitable resource locators include URLs, hot links, file paths, and distinguished names, object class names, table names, and primary database key values, among others.

Bowen, column 4, lines 52-62

In this excerpt, as in the rest of the reference, **Bowen** searches for keywords, not delimiters. One of ordinary skill in the art would not interpret a search for a keyword as searching for a pair of delimiters since one searches for content (keywords) and the other searches for markers of unknown content (delimiters). Further, the claimed invention searches for a <u>pair</u> of delimiters; **Bowen's** searches are not for pairs of keywords. Thus, **Bowen** does not meet the feature of "*scanning programming code for a first pair of string delimiters that are used to delimit text strings*".

Regarding the step of "*determining whether a string within said first pair of string delimiters is a path name to a resource file*", the rejection points to **Bowen's** "*resource file (col. 4, line 56-62 "keyword with resource locators.., include. .URLs, hot links, file paths... among others")...*". This quotation is part of the excerpt shown earlier on this page. As seen there, **Bowen** states that the process it discloses will find keywords, determine their locations, and <u>create an index associating the keyword with its corresponding resource locators</u>. One of ordinary skill in the art would not understand the act of associating a keyword with corresponding resource locators to be the same as determining whether a string <u>is</u> a resource locator. Thus, **Bowen** does not meet the feature of "*determining whether a string within said first pair of string delimiters is a path name to a resource file*".

In the Final Office Action, the Examiner appears to be defending the position that **Bowen** can be read on the steps of claim 1 if the reference shows both the action and a object of the action, even if the context does not show the claimed action being performed in conjunction with the claimed object. For example, **Bowen** shows both scanning (searching) and programming code (programming languages), but the reference does <u>not</u> show scanning programming code.

When Appellants made this assertion previously, the rejection simply pointed to the same cites and asserted that Appellants were making general allegations. It is not a general allegation to assert that searching using programming code is not the same as searching through programming code. Appellants have pointed to specific differences and have not received a reply that addresses the discontinuity between the actions and elements as shown in **Bowen**.

Therefore, as argued above, **Bowen** does not meet the features of claim 1. Contrary to the Examiner's assertion, the arguments presented are not general allegations, but point to specific differences between the claimed invention and the reference cited. Thus, the rejection of this group of claims should be overturned.

A.2. Claims 5, 13, and 21

Claim 5 is representative of this group of claims and recites, "*wherein said path name is a resource bundle*", in addition to the features of independent claim 1. The application defines a resource bundle as follows:

> As stated above Java has developed tools to assist in developing internationalized programs and allowing text strings to appear in the language of the locale. One such tool is the use of resource files commonly referred to in Java as resource bundles. A resource bundle class may be used for externalizing text strings, i.e. not hard-coding strings in the program. The resource bundle class represents a bundle of resources that may be looked up by name. The resources may include appropriate text strings for a given locale that are indexed by what are commonly referred to as keys. Keys are free formatted strings that appear in the program code as well as in the resource bundle thereby allowing the program to access the externalized string. By having resource bundles associated with particular locales, e.g., a resource file with resources associated with the US English locale, a resource file with resources associated with the French locale and so forth, appropriate text strings associated with the particular locale may be loaded at runtime.

Application, page 3, line 16 through page 4, line 3

The rejection cites **Bowen**, column 4, lines 52-62, excerpted above, which refers to resource locators including URLs, hot links, file paths, etc. However, **Bowen** is not concerned with internationalization and has no reason to use or refer to resource bundles, as that term is used in the application. Reading this limitation on **Bowen** ignores the meaning of resource bundles. Thus, the rejection of this group of claims should be overturned.

A.3. Claims 6, 14, and 22

Claim 6 is exemplary of the claims in this group and recites "*wherein said string within said first pair of string delimiters is a path name to said resource file if said string is in a dot delimited notation*", in addition to the features of claim 1. The feature recited in claim 6 is not specifically concerned with the search for delimiters; rather it addresses and further defines the *determining* step by defining how the determination can be made. Against this recitation, the rejection cites column 5, lines 55-64 of **Bowen**, which states:

> As used here, a "keyword" search is a pattern-matching search which tries to locate instances of digital data using a key word or phrase. Many conventional web search engines support keyword searches. Keywords may contain wildcards. For instance, if the question mark is used as a wildcard capable of matching any single character and the asterisk is used as a wildcard capable of matching any zero or more characters, then the keyword "b?t*" would match the words "bat", "bet", "bit", "bot", "but", "battle", "bitten", and "butane", among others. In some cases keywords may also contain regular expressions, such as the regular expressions used in the familiar lexical analysis program lex or the familiar text editors emacs and vi. A keyword may contain smaller keywords connected by operators such as AND and OR.

This citation discusses keyword searching, which does not address the specific recitation of claim 6. Specifically, this excerpt does not address the determination whether a string is a path name. Instead, this excerpt discusses searching and does not address actions subsequent to the search. Since this feature is not shown by **Bowen**, the rejection should be overturned.

Betty Formby

Betty Formby
Reg. No. 36,536
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal are:

1. A method for identifying hard-coded strings that require conversion, the method comprising the computer-implemented steps of:

scanning programming code for a first pair of string delimiters that are used to delimit text strings;

determining whether a string within said first pair of string delimiters is a path name to a resource file; and

if said string is not a path name to said resource file then flagging said string as a possible hard-coded string.

2. The method as recited in claim 1 wherein said string is not flagged as a possible hard-coded string if said string is a path name to said resource file.

3. The method as recited in claim 1, wherein said programming code comprises platform-independent byte code.

4. The method as recited in claim 1, wherein said path name is a uniform resource locator.

5. The method as recited in claim 1, wherein said path name is a resource bundle.

6. The method as recited in claim 1, wherein said string within said first pair of string delimiters is a path name to said resource file if said string is in a dot delimited notation.

7. The method as recited in claim 1, wherein said programming code is scanned line by line until said first pair of string delimiters is identified.

8. The method as recited in claim 7, wherein if there is any more programming code to be scanned after said first pair of string delimiters is identified, then the method further comprises the step of:

continuing to scan said programming code for a second pair of delimiters.

9. A computer program product in a computer readable medium for identifying hard-coded string that require conversion, comprising:

programming operable for scanning programming code for a first pair of string delimiters that are used to delimit text strings;

programming operable for determining whether a string within said first pair of string delimiters is a path name to a resource file; and

programming operable for flagging said string as possible hard-coded string if said string is not a path name to said resource file.

10. The computer program product as recited in claim 9 wherein said string is not flagged as a possible hard-coded string if said string is a path name to said resource file.

11. The computer program product as recited in claim 9, wherein said programming code comprises platform-independent byte code.

12. The computer program product as recited in claim 9, wherein said path name is a uniform resource locator.

13. The computer program product as recited in claim 9, wherein said resource file is a resource bundle.

14. The computer program product as recited in claim 9, wherein said string within said first pair of string delimiters is a path name to said resource file if said string is in a dot delimited notation.

15. The computer program product as recited in claim 9, wherein said programming code is scanned line by line until said first pair of string delimiters is identified.

16. The computer program product as recited in claim 15, wherein if there is any more programming code to be scanned after said first pair of string delimiters is identified, then the method further comprises:

programming operable for continuing to scan said programming code for a second pair of delimiters.

17. A data processing system, comprising:

a processor; and

a memory unit for storing instructions of said processor;

an input mechanism;

an output mechanism;

a bus system for coupling the processor to the memory unit, input mechanism, and output mechanism;

means for scanning programming code for a first pair of string delimiters that are used to delimit text strings;

means for determining whether a string within said first pair of string delimiters is a path name to a resource file; and

means for flagging said string as a possible hard-coded string if said string is not a path name to said resource file .

18. The data processing system as recited in claim 17, wherein the system further comprises:

means for not flagging said string as a possible hard-coded string if said string is a path name to said resource file.

19. The data processing system as recited in claim 17, wherein said programming code comprises platform-independent byte code.

20. The data processing system as recited in claim 17, wherein said path name is a uniform resource locator.

21. The data processing system as recited in claim 17, wherein said resource file is a resource bundle.

22. The data processing system as recited in claim 17, wherein said string within said first pair of string delimiters is a path name to said resource file if said string is in a dot delimited notation.

23. The data processing system as recited in claim 17, wherein said programming code is scanned line by line until said first pair of string delimiters is identified.

24. The data processing system as recited in claim 23, wherein if there is any more programming code to be scanned after said first pair of string delimiters is identified, then the method further comprises:

means for continuing to scan said programming code for a second pair of delimiters.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.